Evidence-Based Computing for Dependability

Yasuhiko Yokote, Ph.D.
Dependable Embedded OS R&D Center
(also with Sony Corporation)

January 22, 2010

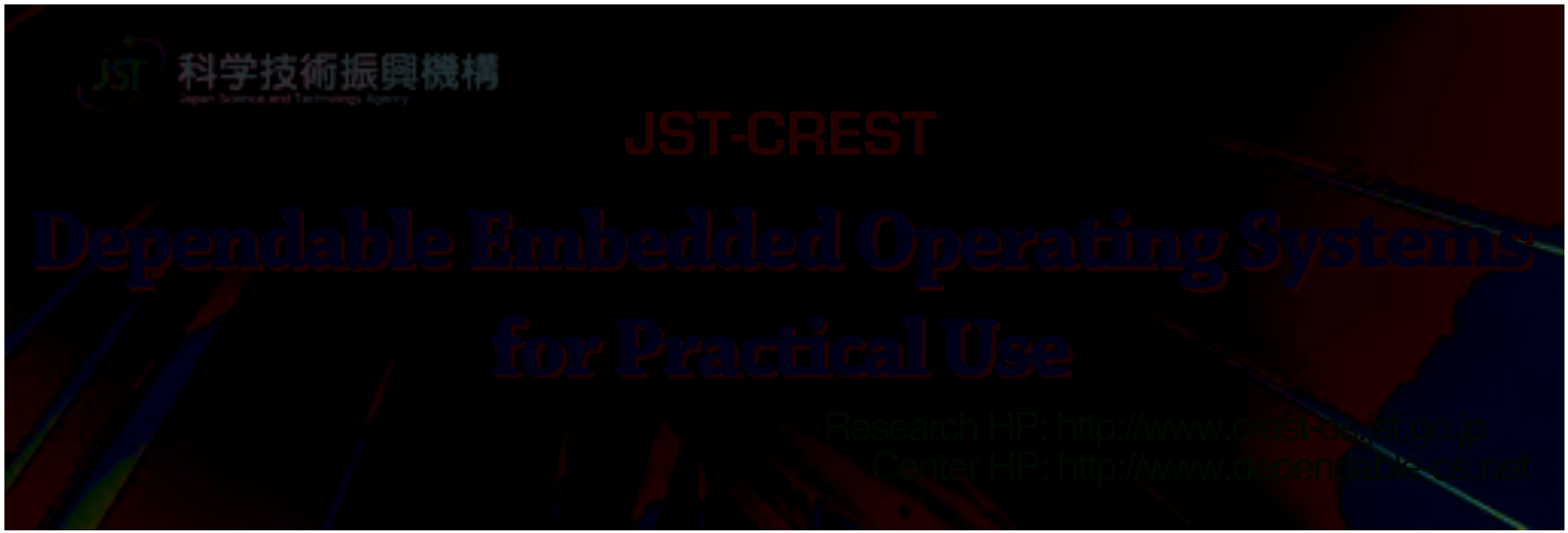
科学技術振興機構
Japan Science and Technology Agency

Agenda

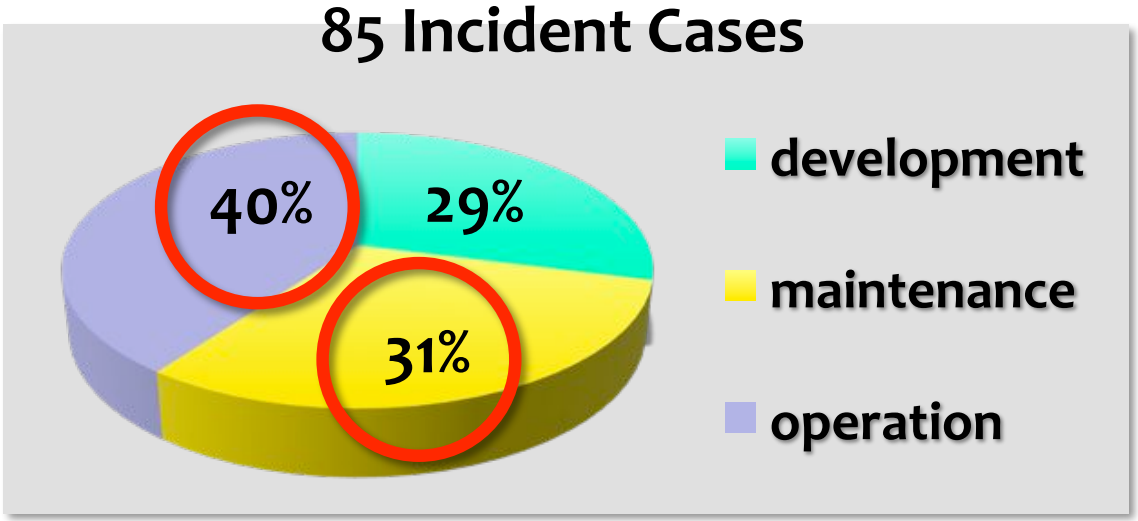
- Case Study
  - Several incident cases have been investigated in terms of what services could be useful to avoid a failure and to minimize impact on it.
  - What could be realized if open systems dependability would be secured?
- Evidence-Based Computing
  - How evidences are managed to sustain Open Systems Dependability.
- D-fops: D-Framework for Open Systems
  - A reference implementation of evidence-based computing.
- Summary

Joint Work with…

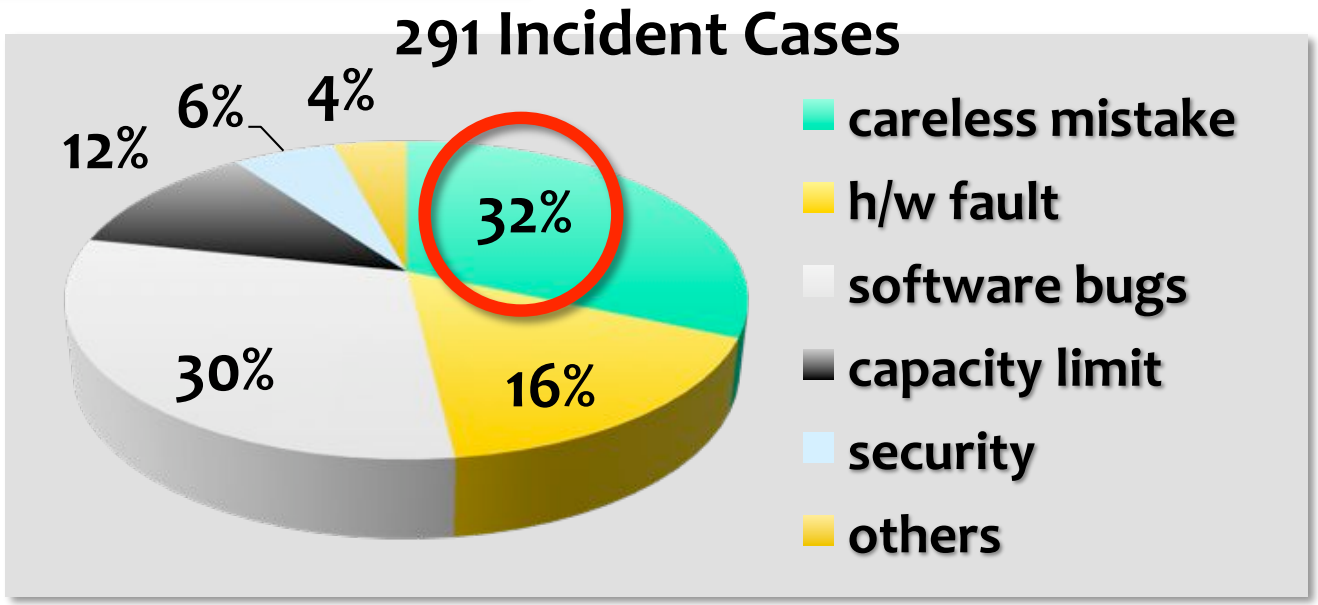
- Asai, Nobuhiro/JST
- Kuramitsu, Kimio/YNU
- Matsubara, Shigeru/JST
- Matsuno, Yutaka/AIST
- Miyahira, Tomohiro/JST
- Nakazawa, Jin/Keio U.
- Ono, Kiyoshi/JST
- Sekiba, Jiro/JST
- Sugaya, Midori/JST
- Takamura, Hiroki/AIST
- Takeda, Shingo/JST
- Yashiro, Makoto/JST



From Two Reports



(source: IPA/SEC Internal Report)



(source: Nikkei Computer 2009.08.19)

Recent Trend Derived from Case Analysis

Large Scale, Complexity

- Increase code size, # of functions, and project scale.
- Exceed project manageability.

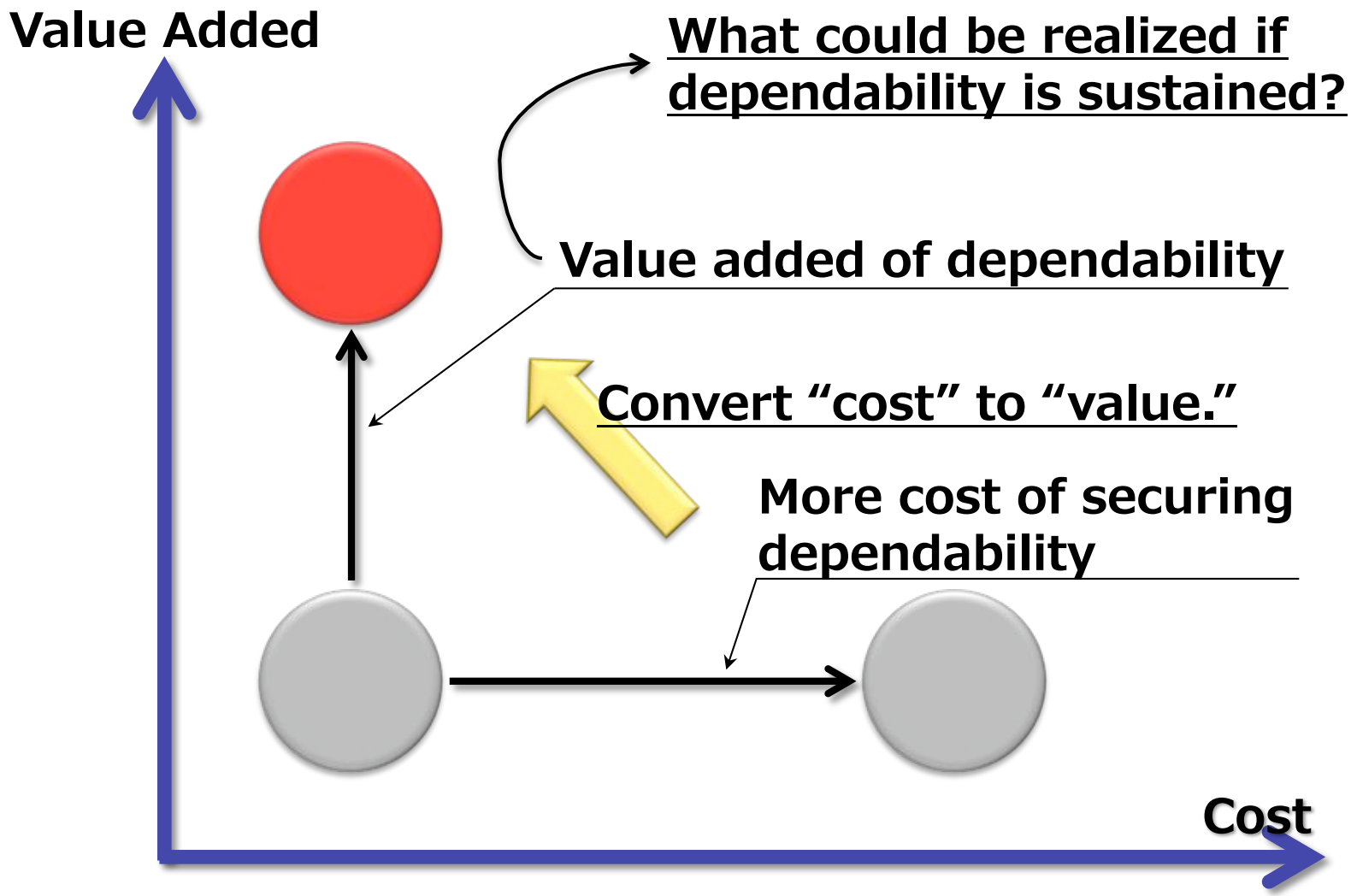
Faster Environment Change

- Reduce innovation cycle.
- Get hard to fix a cause of failures within environment change.

More Stakeholder's Voices

- Change her/his needs frequently.
- Tailor made requirements.

Value of Dependability



Another Market Trend

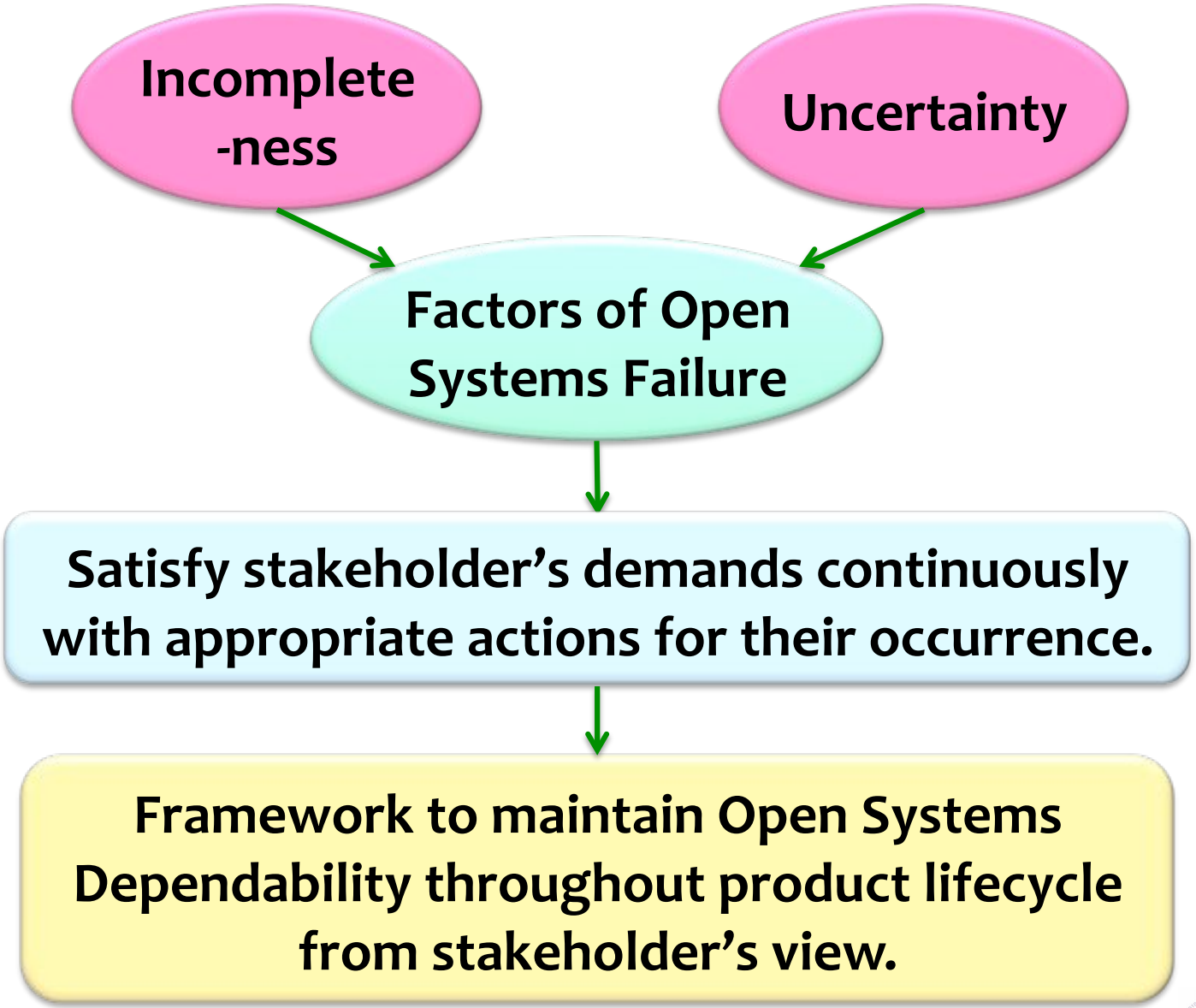
- A profit source is changed due to “free” content.
  - Once digitalized makes it free.
  - No sell-buy direct relationship.

It leads us to:

- Commoditization of OS such as Android/Chrome.
- Value shifts to massive cluster of computers.
- Enrich value with collective knowledge.

☞ It is a good time to reconsider how cost for dependability is paid.

Open Systems Dependability



Value Added – Merits to Stakeholders

Account

- Agree on dependability mutually among stakeholders.
- Account for a cause with its evidence.

Avoid

- Avoid an occurred failure and continue services.
- Apply a workaround.

Manage

- Minimize an impact of a failure.
- Facilitate quick recovery of a failure.

Sustain

- Repeat no same failure.
- Improve service level throughout product lifecycle.

Good News

Concentrate on adding value of products and services from stakeholder's view.	Cost for dependability is converted to product value added.
Take an action to minimize impact on a failure, being agreed with stakeholders.	Conflict with stakeholders is avoidable.
Provide quick repair service for users because of product damage's evidence.	Capturing voices in market makes impact on "recall" minimized.

However, bad news is, due to its nature, a system requires new computing to manage factors of open systems failure.

New Computing to Secure Open Systems Dependability

- Computing itself should be reconsidered in order to take responsibility of an account upon occurrence of factors of open systems failure.

☞ **Evidence-Based Computing**

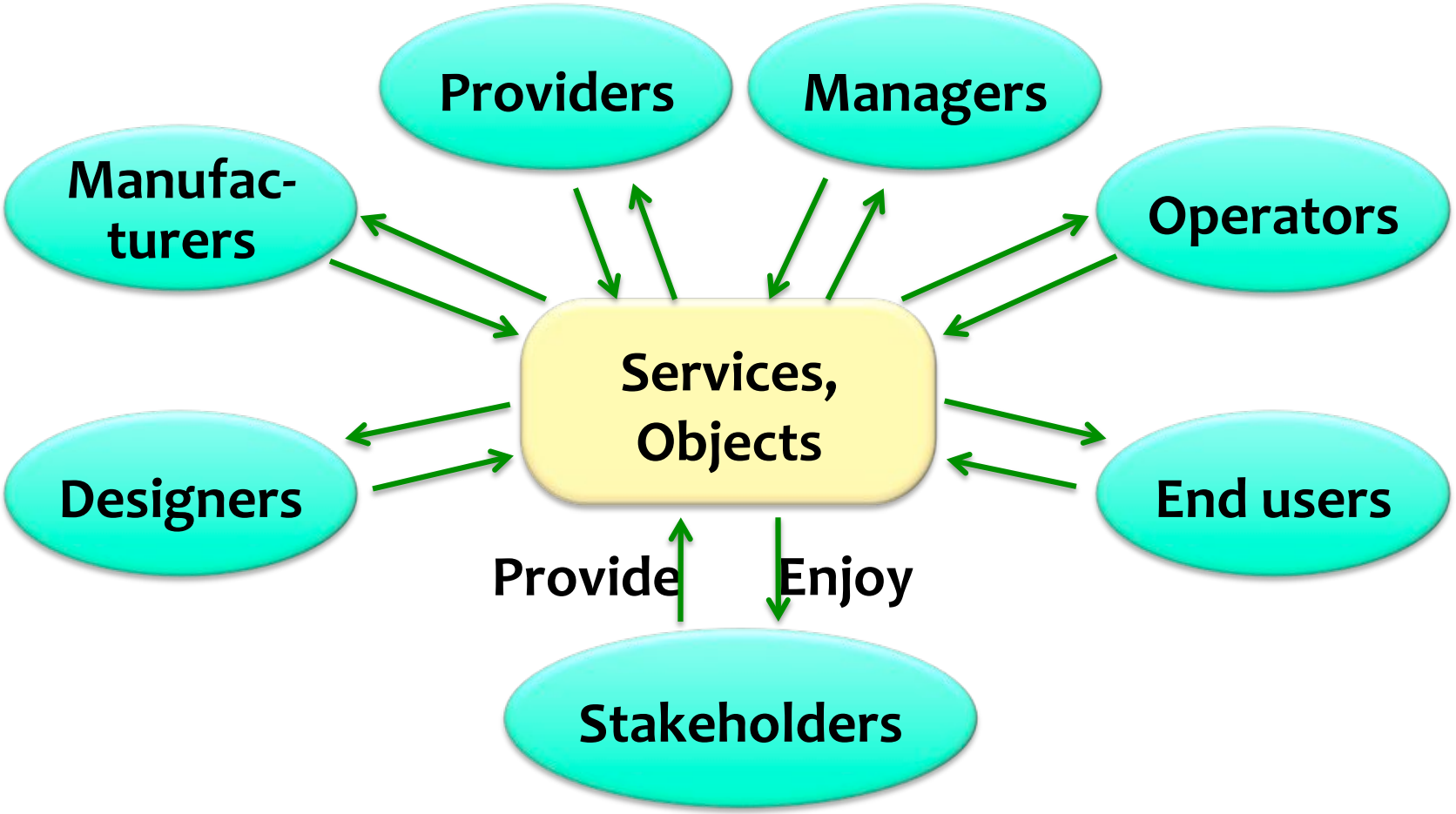
- It is important to;
  - make stakeholders satisfied with restoring to normal,
  - present evidences to stakeholders securely, and
  - create valued product and services because of dependability.

What is “Evidence”?

- It is under discussion in the DEOS project.
- It is a ground to say about something to be true.

  For example,
  - upon occurrence of a failure;
    - history by the time of the failure,
    - records of causes related to the failure,
  - on implementation of functions;
    - records as the functions are implemented based on their requirements,
    - benchmarks as non-functional requirements are secured,
  - on system operation;
    - records as operation is performed according to its operational manual,
    - records as programs are properly deployed.

What is Stakeholder?



These relationships are changing in the product lifecycle.

Ability to Manage Open Systems Dependability

Account for System's Behavior

explain it based on "evidence"

Change System's Configuration

provide a policy upon configuration

Manage Component's Dependency

minimize an impact on occurrence of a factor of open systems failure

Describe sustainable Open Systems Dependability in expectation of stakeholders.

Prerequisite for Describing Open Systems Dependability

Requirement	**What are functional requirements as well as non-functional ones such as security, capacity, and operation?**
Agreement	**What requirements are mutually agreed between stakeholders?**
Compliance	**Which agreements are met in each phase of product lifecycle?**
Change History	**What descriptions are changed and recorded in product lifecycle?**

☞ **D-Case is introduced as a tool to describe Open Systems Dependability.** (which detail is presented tomorrow.)

System Services to Secure Open Systems Dependability

- Several incident cases have been investigated in “what-if” manner…
  - What system services could let occurrence of a failure be avoided if a system provided these services?
  - Several cases have been investigated including;
    - A bug of DoCoMo Cell Phone,
    - Shutdown of Google Gmail services,
    - Unable to check-in with ANA flights, *etc.*

- As a result, 7 system services are introduced, as well as 3 services are added for convenience.

Case Study

- Date: October 12, 2007
- Problem
  - Ticket gates that have IC card reader in several locations in Tokyo area were not operational.
- Cause
  - There was a primitive error in a logic that break big data into small chunks during transmission of some essential information from a sever to ticket gates. This caused infinite retry loop in receiving data in a ticket gate side.
- What system support is required to avoid this problem?
  - Tolerance for configuration change to support new services.
  - Time-shift testing so that a bad situation could occur before a production service.
  - Software anti-aging to avoid firing the accident.
  - Isolation of a failed component to avoid the total failure.
  - Presentation of evidences to identify the root cause.
  - Undo mechanism to shorten down time.

System Services to Avoid Failures in Eight Cases 1

- Evidences.
  - All cases spent a long time to indentify the root cause of their failures.
  - Facilities to probe internals of each component and to record it securely.
- Isolation.
  - Since a single failure leads the system to the total failure in most cases, in which the failed component is not isolated from other systems to avoid it.
  - Facilities to isolate a failed component and to continue other part of services.
- Time-shift testing.
  - If we could do a rehearsal with the same condition of when that incident occurred, e.g. set the system clock to the specific date and time, most cases revealed failures that could happen in the production service.
  - Facility to run a system at a specific time.

System Services to Avoid Failures in Eight Cases 2

- Software anti-aging.
  - Bugs are sometimes fired after long run of the system due to software aging.
  - Facility to avoid software aging including rejuvenation (such as restarting a system).
- Undo.
  - Most failures occurred after system changes. Those failures could be avoided if those changes did undo.
  - Facility to revoke a previous action.
- Proactive management.
  - Some systems reported some errors prior to the incident.
  - Facilities to predict the failure from those errors and take some actions to avoid it.
- Quota.
  - Some cases indicate that congestion condition due to overflow caused the total failure.
  - Facility to limit resource usage.

Summary of Case Study

	DoCoMo	Gmail	ANA Check-In	Tokyo Stock Ex.	Nippon Signal	NTT IP Phone	ANA Ticketing	Air Traffic Control
Evidence		✓	✓		✓	✓	✓	
Isolation	✓			✓	✓	✓	✓	
Time-Shift		✓	✓		✓			✓
Anti-aging				✓	✓			✓
Undo		✓			✓	✓		✓
Proactive man.			✓	✓			✓	
Quota		✓				✓	✓	

In addition to 7 Services…

- Configuration change.
  - Those services require to change system configuration.
  - A policy is given when configuration changes.
- Policy management.
  - An idea is taken from medical treatment.
  - A policy description requires computer assistance, i.e. tools are provided.
- D-Case management.
  - How is Open System Dependability described is standardized.
  - Such a description is assisted by tools such as D-Case editor to avoid ambiguity.

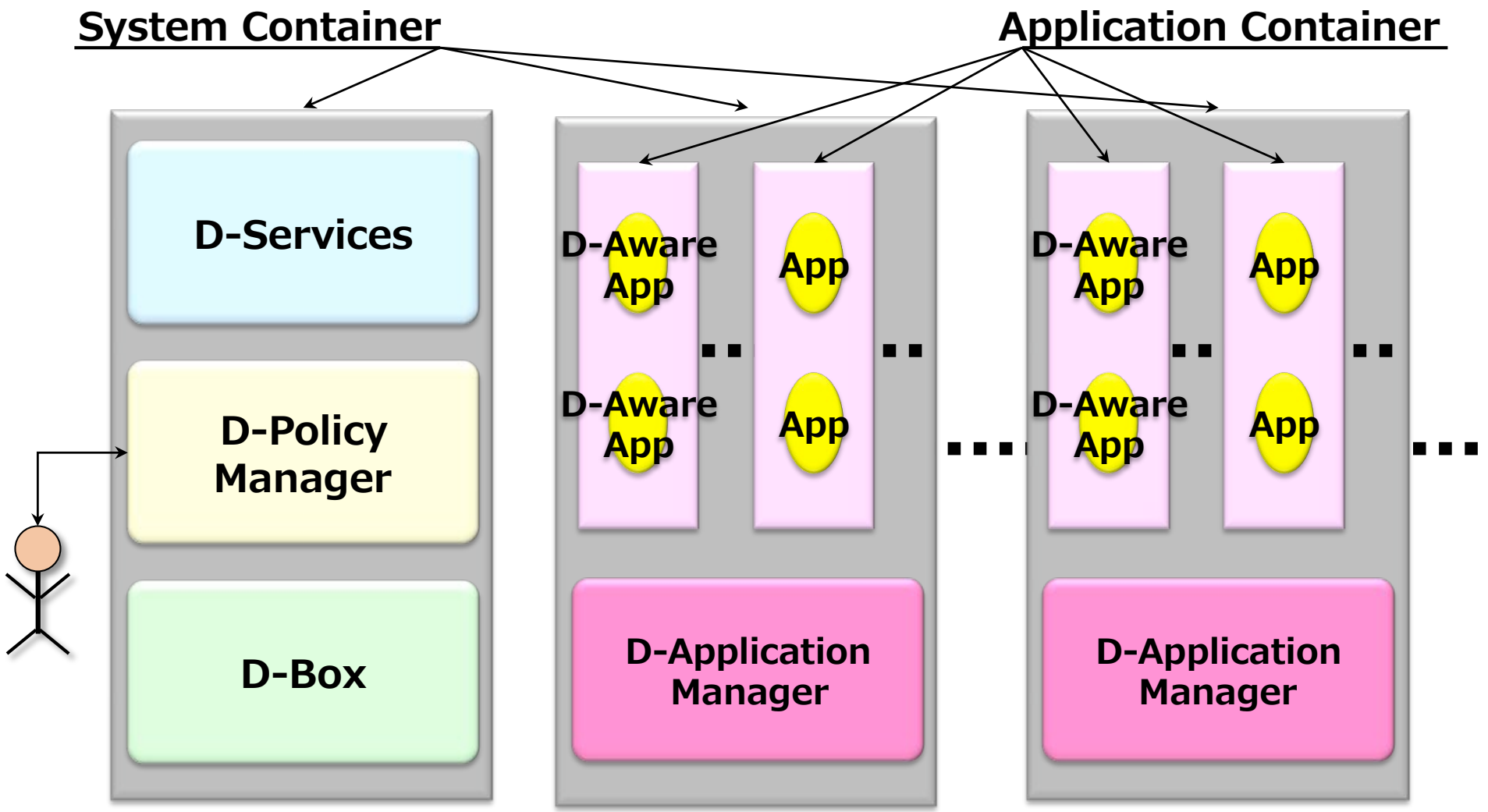
D-fops: Framework for Open Systems Dependability

- D-fops is a reference implementation, which enables us to develop products and services sustaining Open Systems Dependability using 10 system services.

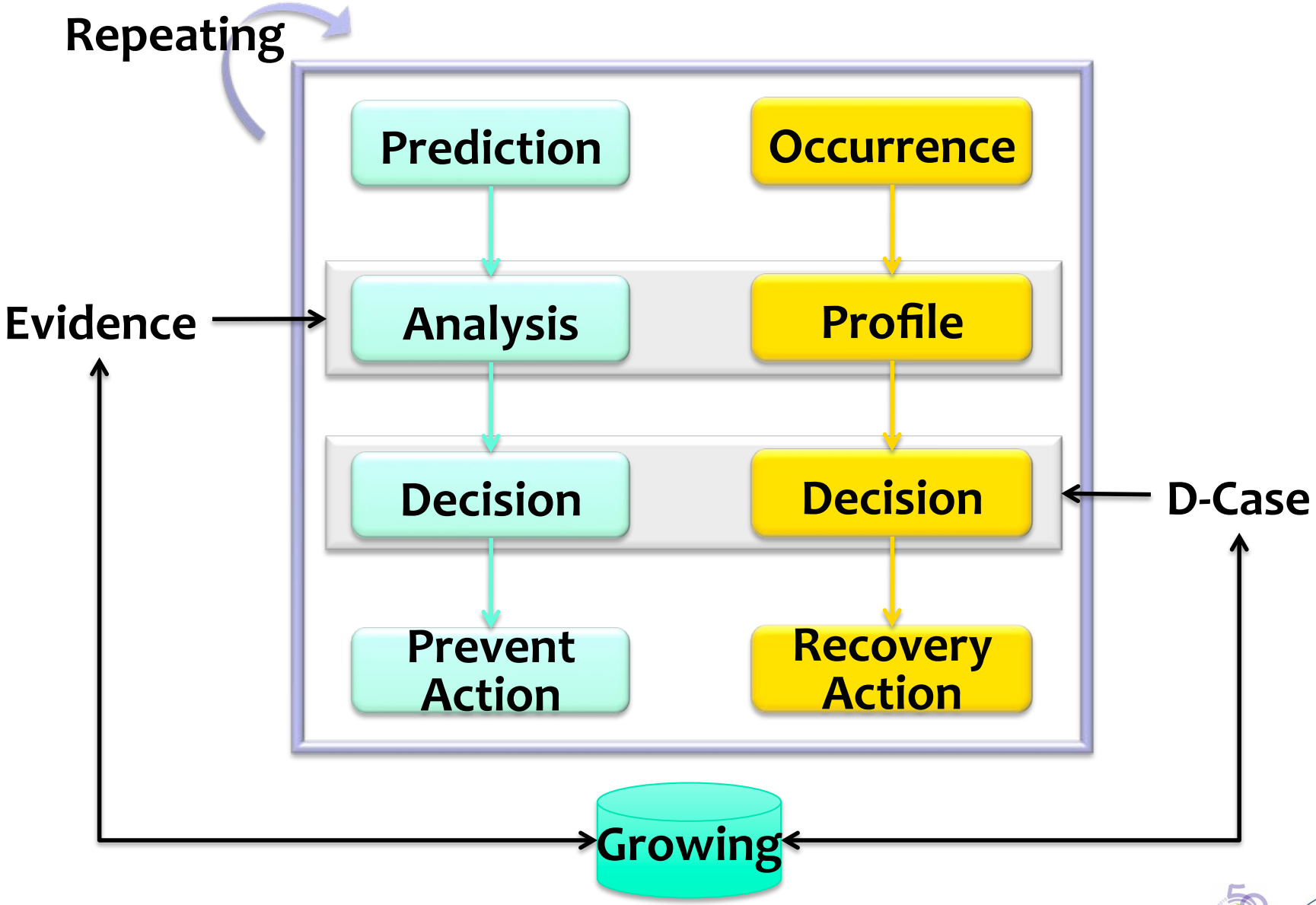
- Using D-fops enables us to embed Open Systems Dependability in products and services throughout their lifecycle.

- D-fops provides tools to facilitate product development as well as its operation and maintenance sustaining Open Systems Dependability.

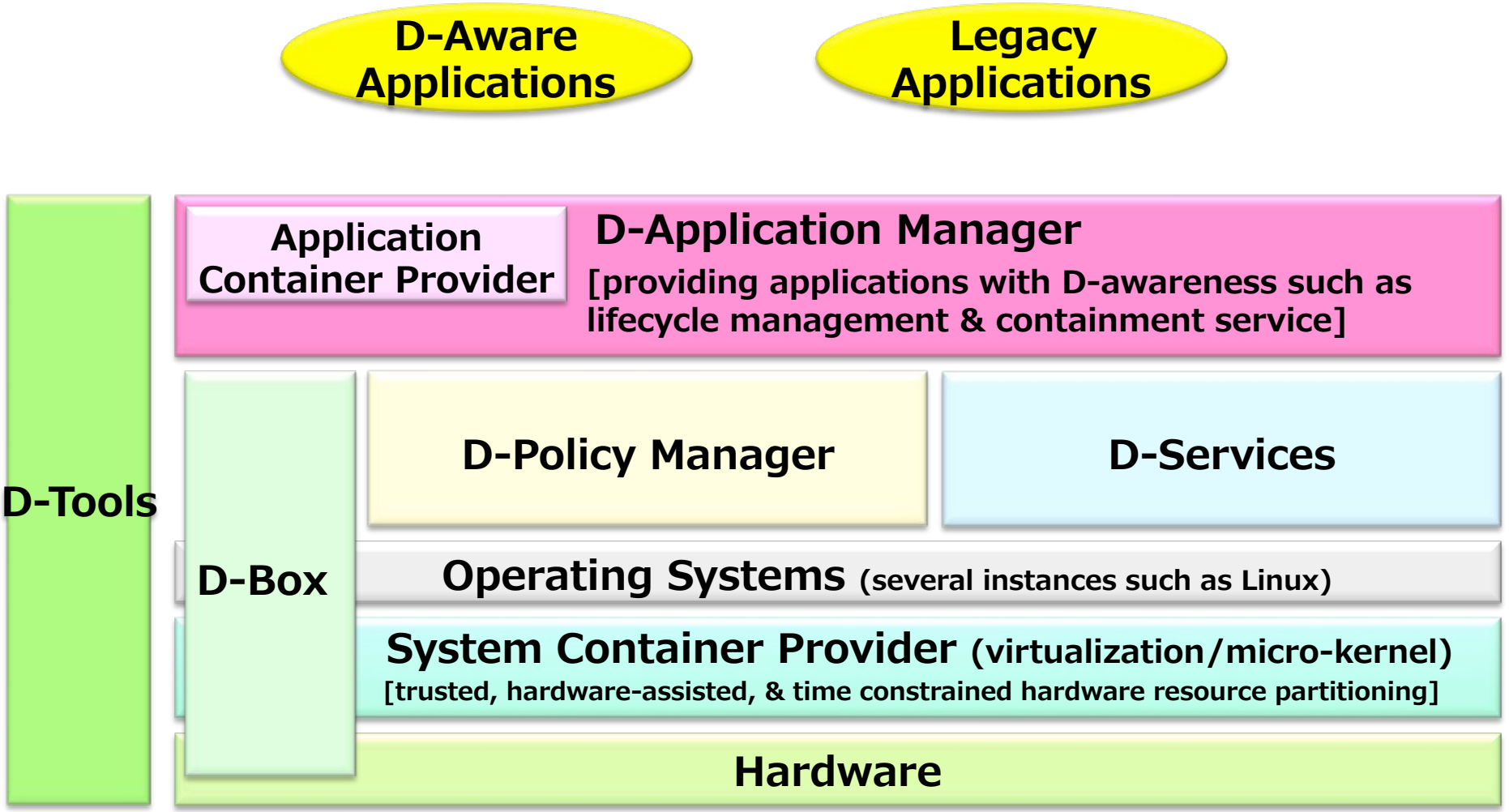
Architecture Overview



Basic Flow of Control



Software Structure

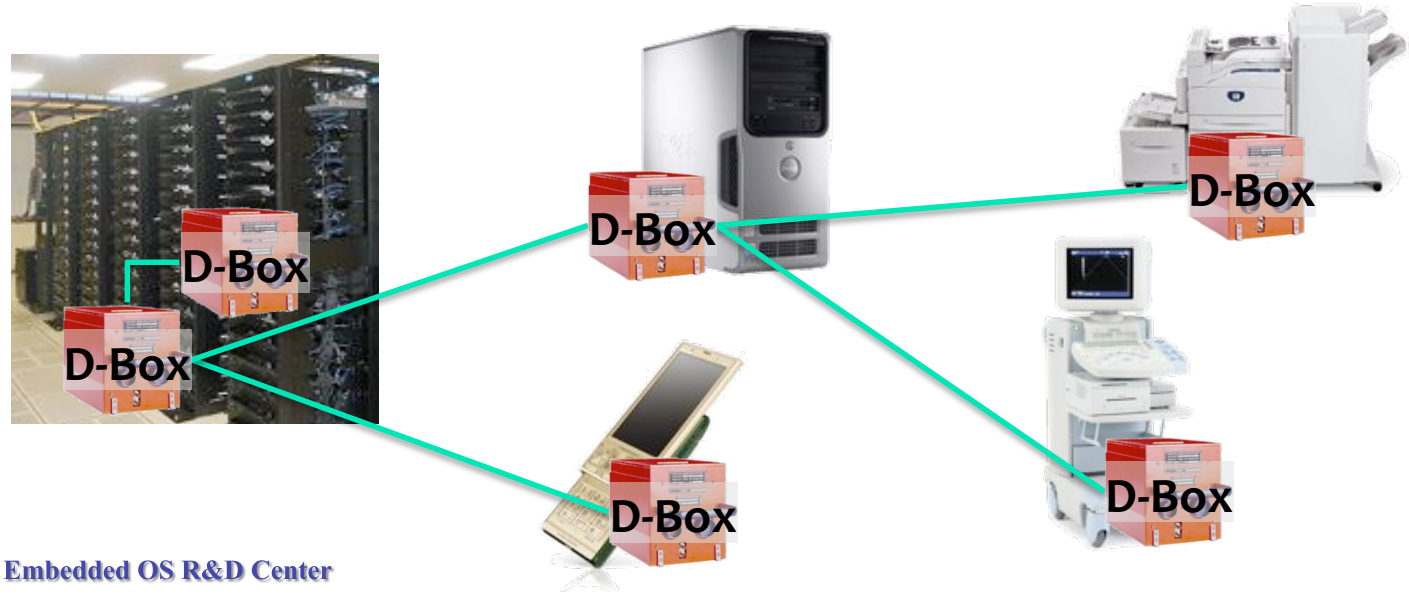


D-Box – Securing Evidences

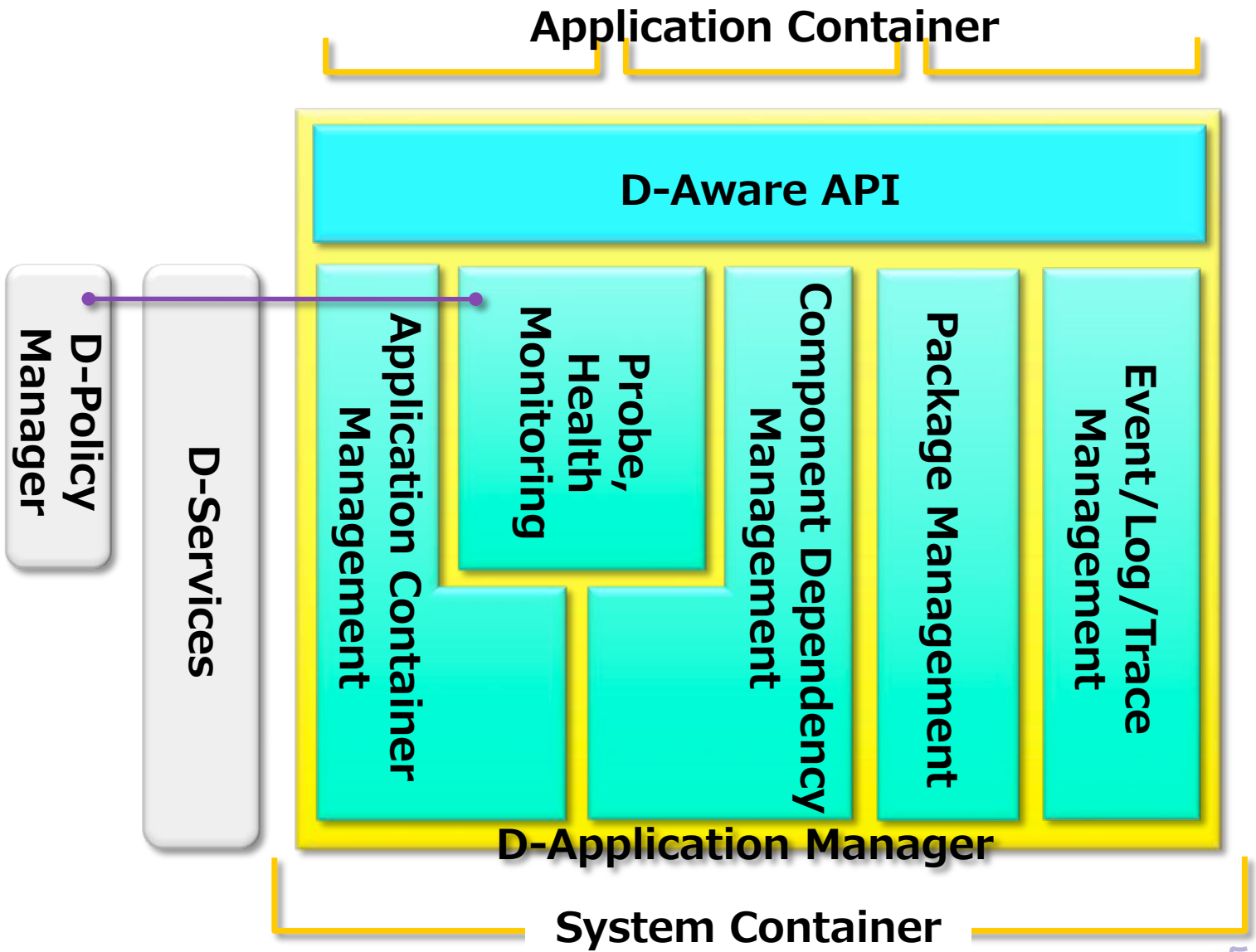
- D-Box is a heart of Evidence-Based Computing, where
  - it is a place to store evidences securely,
  - one D-Box is placed at a device,
  - a chain of D-Box can be structured.
- Access to D-Box is secured, where
  - an end-point is authenticated, and
  - transmission between an end-point and D-Box is encrypted.
- Examples of D-Box content are:
  - system configuration,
  - event/log records,
  - time-stamp, and
  - key files.
- D-Box is similar to a flight recorder.

Network of D-Boxes

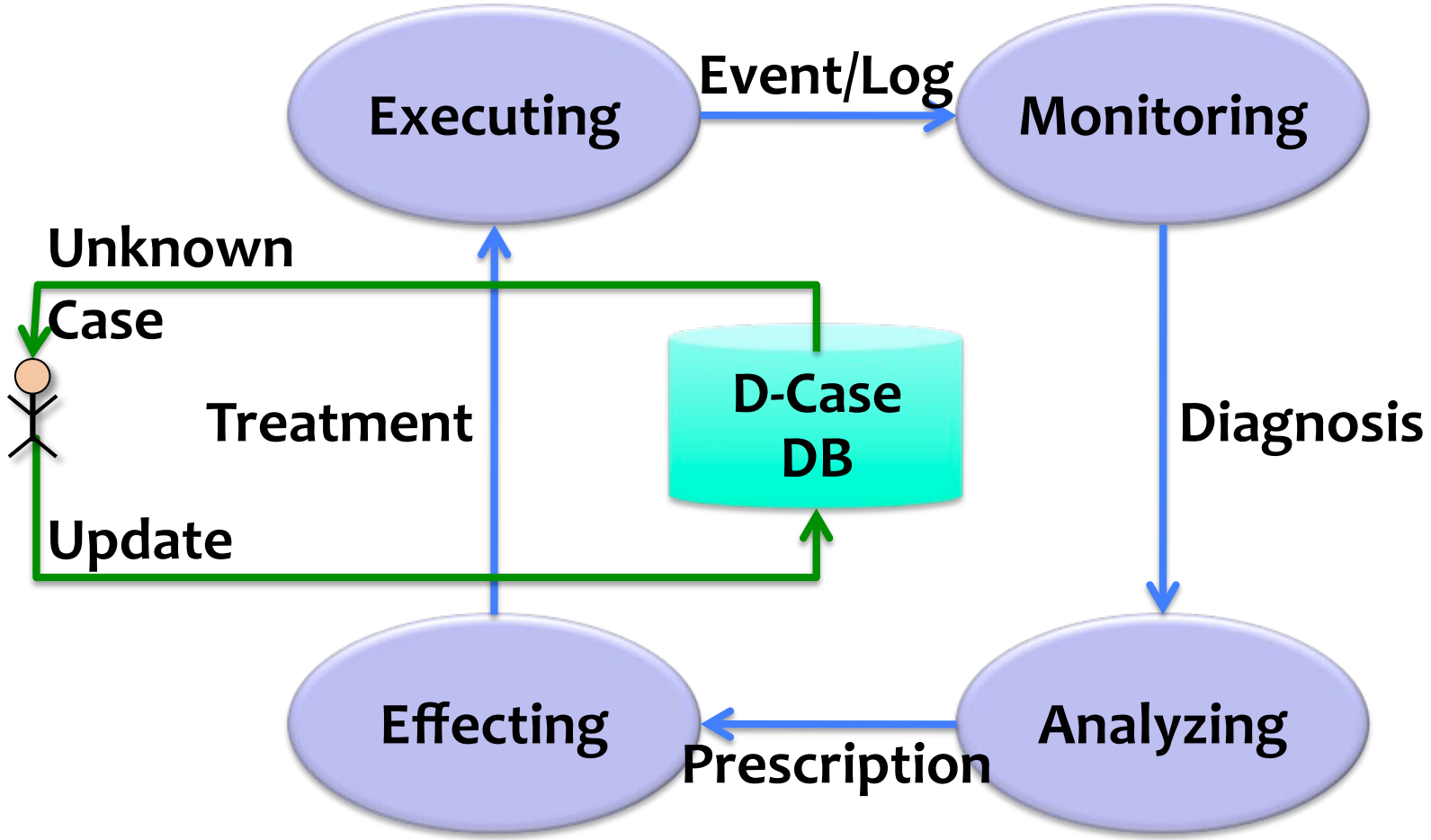
- Network of D-Boxes is representing a dependability chain, i.e. Open Systems Dependability is sustained in a system consisting of linked D-Boxes.
- Each D-Box contains evidences indicating its associated device is secured by Open Systems Dependability.
- Exchanging evidences between D-Boxes enables us to know a “trend” of dependability.
  - sample use case: remote maintenance



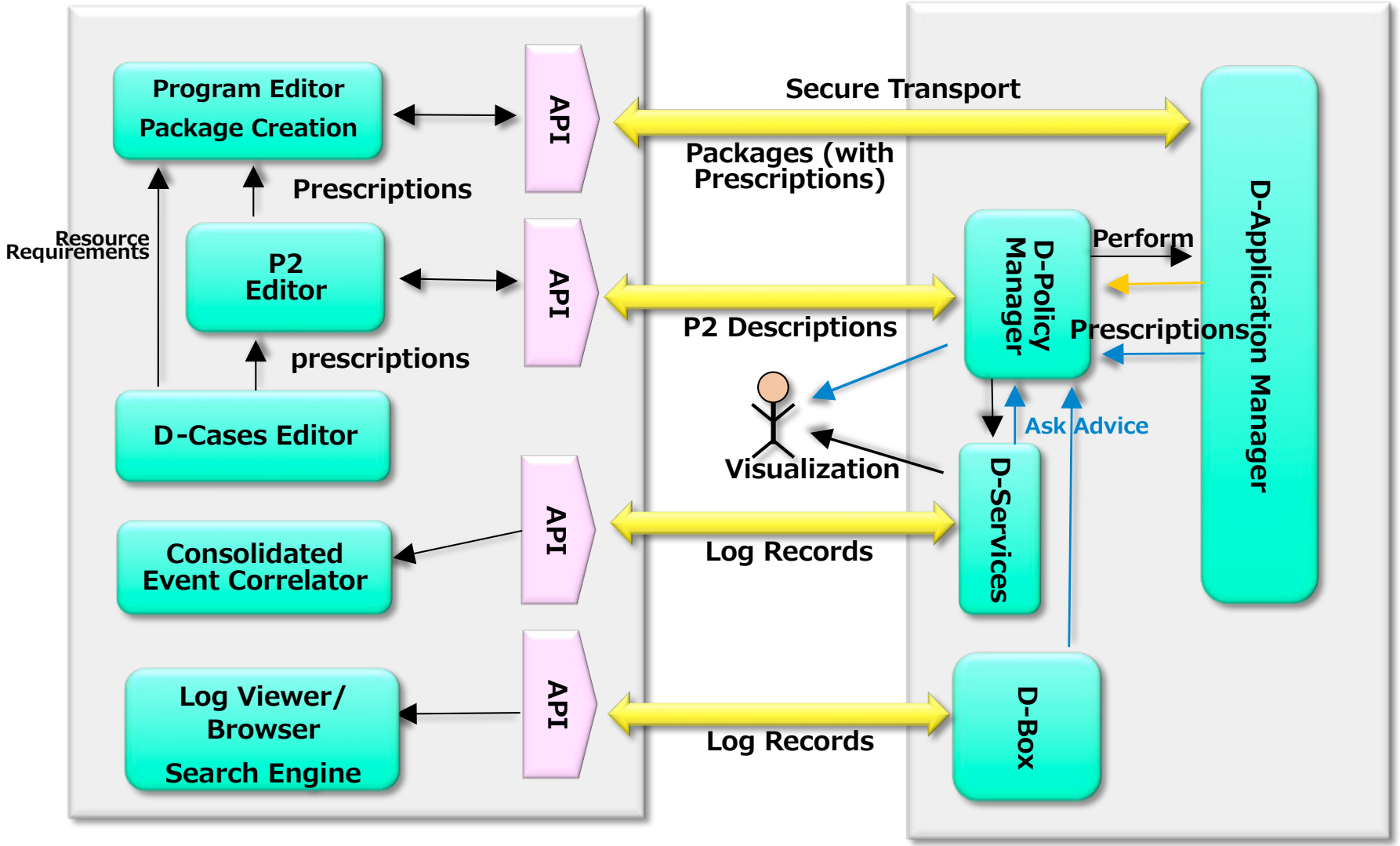
D-Application Manager



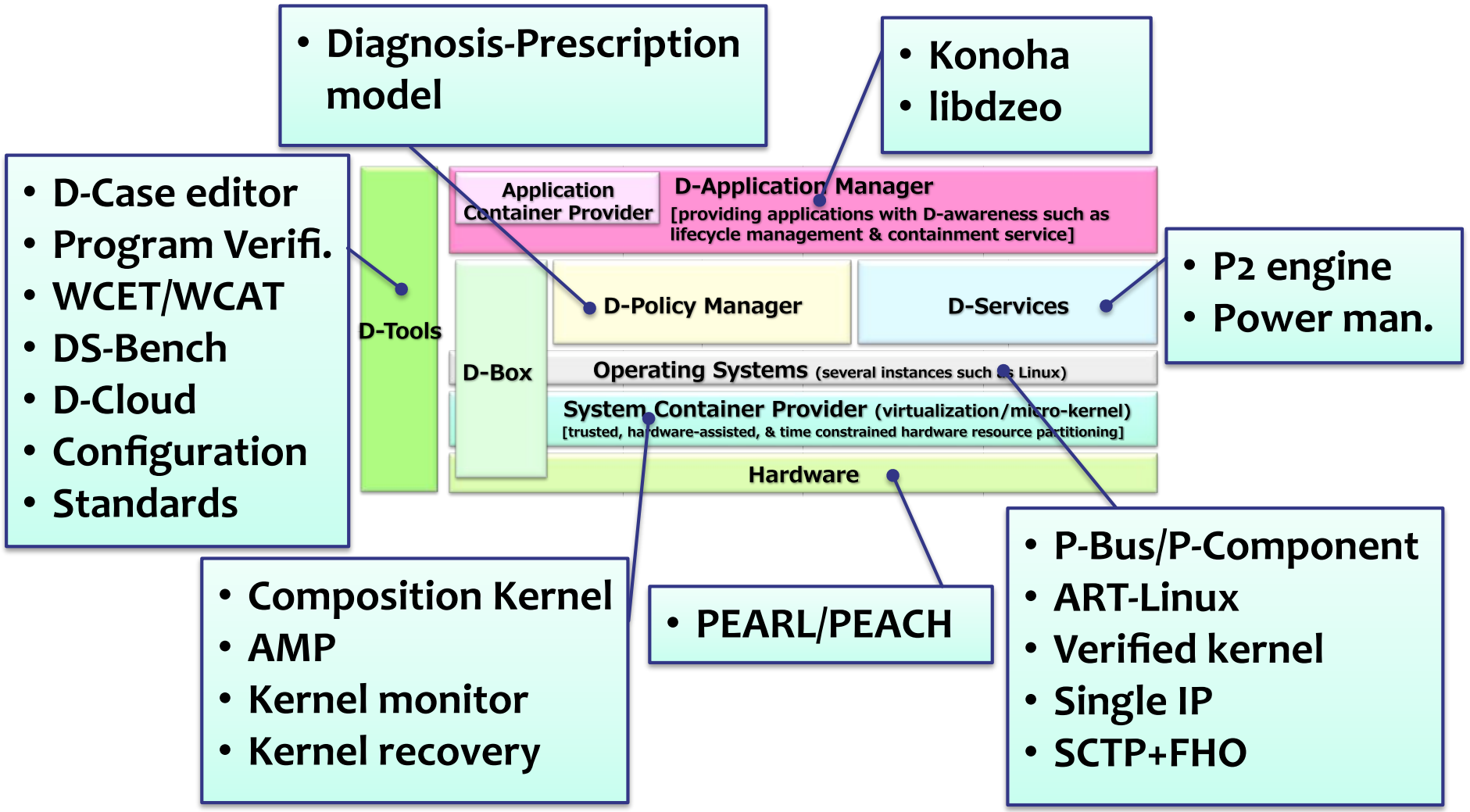
System and D-Case in Loop



D-Case Tools

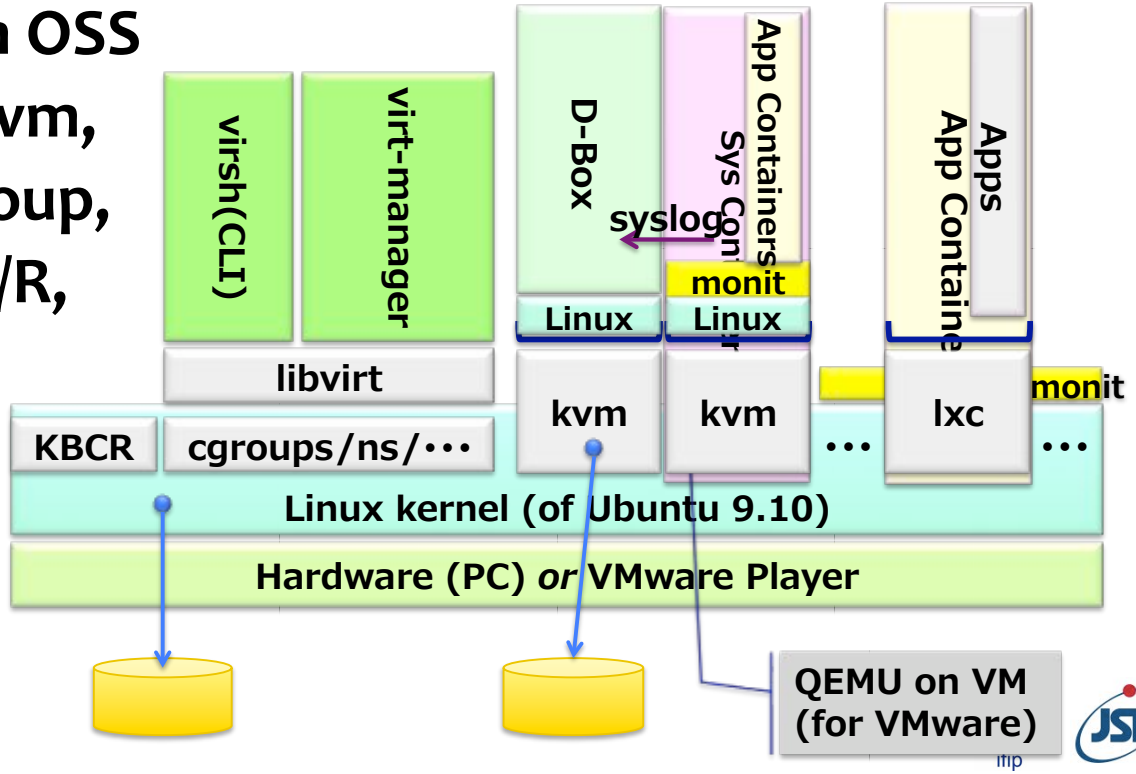


Technology Map of the DEOS Technologies for D-fops



D-fops Prototype

- Feasibility study
  - The D-fops prototyping is underway using OSS.
  - The first implementation utilizes OSS as much as possible to investigate what can be done with OSS.
  - The succeeding implementation will integrate elemental technology from each research group.
- Components from OSS
  - libvirt, qemu-kvm, monit, lxc, cgroup, kernel based C/R, upstart, etc.



Lessen Learned from Prototype

- OSS is utilized in a specific environment where that OSS was developed.
  - It is difficult to use it in a practical use case, i.e. it is necessary a lot of adaptation.
  - It is worth to develop a Framework to integrate necessary components into it securing Open Systems Dependability.
- Some functionalities are missing in the DEOS Project.
  - Examples are D-Box, application container, component dependency management, software anti-aging, etc.
  - Capturing symptoms of a failure is difficult in practical use case.

Summary and Open Questions

- In order to prepare occurrence of factors of open systems failure;
  - Evidence-Based Computing has been proposed.
  - D-fops is implementing as its reference implementation.
  - Products and services embedded with D-fops enable us to create a new value to stakeholders.
- Open Questions…
  - What “evidences” could be?
  - What is a best way to participate stakeholders in securing dependability?
  - What “change” could be anticipated by securing Open Systems Dependability?

More Information

http://www.dependable-os.net/index-e.html

White Paper and Video demo

Thank you!